ПРЕДСТАВЯНЕ НА ЗНАНИЯ С ПОМОЩТА НА ЕЗИКА KRL

KRL (Даниел Бобров, Тери Виноград) е език за представяне на знания, създаден през втората половина на 70-те години на 20-ти век (основната статия за KRL е публикувана през 1977 г.).

ОБЩА ХАРАКТЕРИСТИКА НА ЕЗИКА KRL

KRL е декларативен език за представяне на знания. Знанията в KRL се организират като множество от понятия (концептуални единици, conceptual entities) и съответните им описания. В системите за програмиране на основата на KRL се извършват три основни операции:

- търсене на обекти (понятия), които са съпоставими с дадено описание;
- съпоставяне на две описания с цел установяване дали те са сравними при конкретни условия (за текущите цели на работата на системата);
- разширяване на дадено описание с цел обхващане на нови знания (приспособяване на дадено описание за целите на описанието на нови знания).

МНОГОСТРАННИ ОПИСАНИЯ НА КОНЦЕПТУАЛНИТЕ ЕДИНИЦИ

Всяко описание се състои от един или няколко дескриптора. Например, описанието на даден обект от някаква сцена може да включва дескриптори, съответни на: "предметът, който е съседен на масата", "нещо, изработено от дърво", "нещо, боядисано в кафяво", "един стол" и др. Някои от тези дескриптори означават факти, които могат да бъдат разглеждани като допълнителни твърдения за съответните обекти, докато други дескриптори служат за означаване на различни гледни точки за описание чрез сравнение.

Така всяко описание на дадено понятие (обект) може да бъде комбинация от различни типове описания (дескриптори), които включват:

- осигуряване (избор) на уникален идентификатор (избор на подходящо име като например "Boston");
- присъединяване (причисляване) на даден обект към определена категория (например: "is a city");
- означаване на ролята на обекта в някакъв сложен обект или събитие (например: the "destination" of a particular trip, т.е. "крайната точка" на дадено пътуване);

- означаване (формулиране) на релация, в която обектът участва (например: being "to the north of Providence");
- включване на (сложна) логическа формула, която има стойност "истина" за дадения обект (например: "Either this person must be over 65, or a widow or widower with dependent children.");
- описание на дадения обект в термините на принадлежност към определено множество или описание на дадено множество чрез обектите, от които се състои то;

- комбиниране на някои от горните типове дескриптори в описания, зависещи от времето (time-dependent descriptions), например: "the place you are today".

При създаването на съответното множество от дескриптори потребителят се ръководи от интуитивната си представа за това, как те ще се използват в процеса на логически извод. Авторите на езика KRL смятат, че избраният от тях подход е по-полезен, отколкото стандартните, унифицирани декларативни подходи за представяне на знания (какъвто е например предикатното смятане от първи ред).

ОПИСАНИЯ, ОСНОВАНИ НА СРАВНЕНИЕ С ДРУГИ ОБЕКТИ И ПРОТОТИПИ

В езика KRL особено се набляга на възможността даден обект да се опише чрез съпоставяне с други, вече известни (описани) единици - обекти, понятия и др.

Обектът, който се използва като основа (база) за сравнение, се нарича прототип. Той осигурява съответна перспектива, от гледна точка на която се описва другият обект. Детайлите на сравнението могат да бъдат разглеждани като по-нататъшна спецификация на прототипа. По принцип е напълно възможно (и авторите на KRL го намират за естествено) един обект да бъде описан в рамките на дадена система от знания единствено чрез множество от такива сравнения.

Тази идея обикновено не се осъществява в чист вид при използване на езика KRL, но така или иначе тя е основната разлика между идеологията на KRL и идеологията на стандартните декларативни (логически) представяния, основани на формули, които са съставени (изградени) от примитивни предикати.

Особености на описанията, основани на сравнение:

- При описанието на даден обект чрез сравнение обикновено като прототип се използва не конкретен (друг) обект, а някакъв стереотип, който представя типичния обект от даден клас. Такъв прототип има описание, което може да не е валидно (вярно) за нито един конкретен обект от съответния клас, а комбинира знанията, които са верни по премълчаване (подразбиране) за обектите от този клас при отсъствие на конкретна (специфична) информация.

- Един обект (или събитие) може да бъде описван(о) по отношение на различни прототипи, като за целта бъде извършвана по-нататъшна специализация на съответните перспективи. Например, фактът "Last week Rusty flew to San Francisco" може да бъде изразен чрез описание на събитието като типичен екземпляр на понятието "пътуване" (Travel) с превозно средство, специфицирано като самолет, крайна точка (цел) - Сан Франциско и т.н. Същевременно, този факт може да бъде разглеждан и като екземпляр на понятието "посещение" (Visit) с изпълнител – човек на име Ръсти и т.н.

- По-нататъшното уточняване (specification) в едно описание чрез сравнение може да продължава или чрез конкретизация на някои по-малко специфични свойства на прототипа, или чрез противопоставяне на някои особености на прототипа, които се смятат за верни по премълчаване (ако няма по-конкретна информация за тях). Възможно е да се извършва сравнение и с някой конкретен обект - прототип, а не само с абстрактни (общи) прототипи, например: "He is like Brian, but shorter and with red hair".

ВЪВЕДЕНИЕ В СТРУКТУРАТА НА ОПИСАНИЯТА И СИНТАКСИСА НА ЕЗИКА

Данните в KRL се описват с помощта на специални структури, наречени елементи (units). Всеки елемент (unit) включва множество от описания (descriptions) и служи за означаване на даден обект, понятие и т.н. Той има уникално име и е причислен към определен тип (category type). Всеки елемент (unit) съдържа определен брой слотове (slots), които имат свои имена и включват описания на различни характеристики на съответния обект (понятие).

Слотовете се използват и за описание на тези подструктури на елемента, които са важни за процеса на сравнение. Всеки слот има свое уникално име. Слотът със служебно (специално) име SELF служи за описание на обекта, който се означава от съответния елемент (unit). С всеки слот може да бъде асоциирано и множество от процедури, които се активират при определени условия в процеса на използване на елемента.

Забележка. В нашите примери ще използваме следните съглашения:

- ще използваме главна буква в началото на името на всеки елемент (unit);
- имената на слотовете ще означаваме с малки букви.

Всяко описание представлява списък от дескриптори, а всеки дескриптор съдържа множество от съответни характеристики (features). Могат да се използват фиксиран брой типове дескриптори (в литературата се говори за 12 типа дескриптори). Например, типът дескриптор, който служи за описание чрез сравнение, се нарича перспектива. Всяка перспектива в KRL се изразява по следния начин:

```
(a <prototype> with
     <identifier-1> = <filler-
                      description-1>
       ...
     <identifier-n> = <filler-
                      description-n>)
```

Тук <prototype> е името на елемент (unit), който служи за основа на сравнение, а <identifier-i> е или име на слот на прототипа, или описание, което е съпоставимо с точно един слот на прототипа.

Пример:

```
(a Person with
   name = "Joe"
   (an Address) = "104 Main
                   Street")
```

Тук name е името на слот в описанието на елемента Person. Предполага се, че в описанието на Person има точно един слот, който е съпоставим с дескриптора (an Address).

На следващата фигура е показано примерно описание на пътуването на Ръсти до Сан Франциско чрез средствата на езика KRL. Най-общо синтаксисът е подобен на този на езика Лисп. Квадратните скоби [...] ограничават описанието на всеки елемент (unit). Ъгловите скоби <...> ограничават описанията на отделните слотове. Всеки съставен дескриптор се ограничава от кръглите скоби (...). Фигурните скоби {...} се използват за комбиниране на няколко дескриптора в едно описание.

Следва описанието на "Rusty's trip to San Francisco" със средствата на езика KRL:

```
[Travel UNIT Abstract
        <SELF (an Event)>
        <mode (OR Plane Auto Bus)>
        <destination (a City)>]
[Visit UNIT Specialization
        <SELF (a SocialInteraction)>
        <visitor (a Person)>
        <visitees (SetOf (a Person))>]
```

```
[Event137 UNIT Individual
   <SELF {(a Visit with
             visitor = Rusty
             visitees = (Items Danny
                              Terry))
          (a Travel with
             destination = SanFrancisco
             mode = Plane)}>]
```

Пояснения. Travel, Visit и Event137 са имената на дефинираните в примера unit-и. Те са причислени към различни типове: Abstract, Specialization, Individual. В описанията се цитират и други, вече известни unit-и: Event, Plane, Auto, Bus, SanFrancisco и др. Събитието Event137 е описано чрез сравнение с два различни прототипа - Visit и Travel. С помощта на конструкцията (Items Danny Terry) се означава множество, което съдържа поне Danny и Terry. SanFrancisco е име на unit, описан като City.

ТИПОВЕ ЕЛЕМЕНТИ (CATEGORIES OF UNITS) В KRL

Всеки елемент (unit) в KRL се причислява към някакъв тип (category type), който може да бъде елемент на следното множество: Basic, Abstract, Specialization, Individual, Manifestation (проява), Relation, Proposition.

Abstract, Basic u Specialization

Елементите от тези три типа се използват за описание на категории като Person, Integer и др. Те се използват основно като прототипи за (в) перспективи. Разликата между тях е съществена главно от гледна точка на съпоставянето по образец в KRL (за matcher-а на KRL).

Категориите от тип Basic (базов тип) представят прости непресичащи се части на света (предметната област) чрез различни видове обекти (например: dog, bacteria, ...). Matcher-ът на KRL предполага, че не съществуват индивиди, които принадлежат едновременно на повече от един базов тип.

Специализациите (категориите от тип Specialization) представят различни (по-нататъшни) подразделения на даден базов тип (като например: Poodle, E. Coli и др.). Всеки прототип от тип Specialization (т.е. всеки специализиран прототип) има дескриптори и процедури, асоциирани към тях, които са по-специфични от тези, свързани със съответния базов тип. Често описанието на даден специализиран прототип служи по същество за описание на определен подклас на даден базов тип (клас) или друга специализация.

Абстрактните категории (категориите от тип Abstract) обикновено се използват за описание на абстрактни понятия (например: action, living thing, ...). За тези абстрактни понятия се задават съответни дескриптори и процедури, които могат да се наследяват от обектите, описани чрез перспектива със (по отношение на) съответния абстрактен прототип.

Индивидуални елементи (елементи от тип Individual)

Използват се за описание на конкретни обекти. В процеса на съпоставяне (на работа на matcher-а на KRL) се предполага, че даден индивидуален обект не може да бъде съпоставен с друг (различен от него) индивидуален обект. Определянето на това, кои обекти ще бъдат описани чрез елементи от тип Individual, се извършва конкретно за всяка предметна област в зависимост от типа на решаваните задачи (в зависимост от типа на логическите изводи, които предстои да се извършват).

Елементи от тип Manifestation ("проява")

Използват се с цел групиране в едно цяло на множество от описания, които принадлежат на някакъв (конкретен) обект. Най-често това се прави в случаите, когато конкретният обект, който е носител на някакви (известни) свойства, не е известен (когато този обект се дефинира чрез известните или характерните му свойства).

Пример: криминалните истории, в които е известно, че в къщата има убиец (известни са много от "свойствата" (характеристиките) на убиеца, но не е известно кой (индивид) е убиецът). В такъв случай убиецът може да бъде описан чрез елемент от тип "проява", с който не е асоцииран конкретен индивид.

Релации (relations) и твърдения (propositions)

С помощта на елементи от тип Relation се описват релации (или предикати) като абстрактни отношения. Елементите от тип Proposition представляват конкретни екземпляри (instantiations) на релации (твърдения, свързани със (отнасящи се до) съответни релации). Обикновено истинностната стойност на дадено твърдение се задава в явен вид (а не е логическо следствие от съдържанието на базата от знания).

ТИПОВЕ ДЕСКРИПТОРИ В KRL

Всеки дескриптор в дадено описание на KRL задава определена (независима от другите) характеристика на съответния обект. Всеки от допустимите в KRL типове дескриптори съответства на определен естествен подход за описание на концептуални обекти. В различните типове дескриптори се използват определени ключови думи (като например: a, the, from, which, ...), които се основават на аналогия със съответни прости фрази на английски език.

При описанието на даден (концептуален) обект често се използват различни типове дескриптори, които съответстват на различни аспекти (гледни точки) на това описание. В тези описания, в които се използват различни типове дескриптори, обикновено част от информацията се дублира (т.е. в KRL се работи обикновено с редундантна информация).

Пример. Понятието "ерген" (bachelor) може да бъде описано по няколко начина:

- може да има прототипен елемент Bachelor и конкретен индивид, описан чрез (a Bachelor with ...);

- може да бъде дефиниран едноместен предикат IsBachelor (неговата дефиниция би могла да включва например специална процедура - тест за ергенство). Тогава даден индивид - ерген моге да бъде описан с използване на предиката (which IsBachelor);

- ергенството може да бъде описано косвено (индиректно) с помощта на прототипи MalePerson и Adult и предикат IsMarried например чрез описанието

{(a MalePerson) (an Adult) (NOT (which IsMarried))};

- може да бъде дефиниран елемент (unit) Marriage със слотове "съпруг" (malePartner) и "съпруга" (femalePartner) и да бъде използвано описание от следния вид:

{(a MalePerson) (an Adult) (NOT (the malePartner from (a Marriage)))}.

Следва списък на допустимите типове дескриптори в KRL.

Директен указател (direct pointer)

Формат: име на unit, число, низ,

`<израз на езика Лисп>

Употреба: указател към unit или данни, включен директно в описанието. Служи за означаване на конкретен (уникален) идентификатор.

Примери: **Block17**, **PaloAlto**, **356**,

"a string",

`(A PIECE (OF LIST) STRUCTURE)

Перспектива (perspective)

Формат:

```
(a <prototype> with
   <identifier-1> = <filler-1>
          ...
   <identifier-n> = <filler-n>)
```

Употреба: причислява даден обект към дадена категория. Сравнение на текущия обект с прототипа и по-нататъшно уточняване на този обект чрез конкретизация на някои негови слотове.

Пример:

```
(a Trip with destination = Boston
           airline = TWA)
```

Спецификация (specification)

Формат:

```
(the <slotSpecifier> from <view>
   <targetDescription>)
```

Употреба: определя (специфицира) текущия обект в термините на неговата роля в дадена перспектива или прототип: <view>. Определя някаква роля в комплексен обект или събитие (например: the "destination" в дадено конкретно пътуване (trip)).

Пример:

```
(the age from Person ThisOne)
```

Това е част от описанието на unit с име Traveller:

```
[Traveller UNIT Specialization
   <SELF (a Person)>
   <age {(XOR Infant Child Adult)
         (using (the age from Person ThisOne)
            selectFrom
               (which isLessThan 2) ~Infant
               (which isGreaterThan 11) ~Adult
               otherwise Child)}>
      ... ]
```

Тук Person е unit със следната дефиниция:

```
[Person UNIT Basic
   <SELF>
     ...
   <age (an Integer)>
     ... ]
```

Така в нашия пример спецификацията (the age from Person ThisOne) означава слота age от описанието на този (същия) Traveller като специализация на базовия тип Person.

Предикат (predication)

Формат:

(which <predicateName>.<predicateArgs>)

Употреба: описва релация, в която участва обектът. Начин за описание на даден обект в термините на някаква релация и нейните аргументи. Позволява включване на специални процедури.

Примери:

```
(which Owns (a Dog))
(which IsBetween Block17 (a Pyramid))
```

Булев дескриптор (logical, boolean)

Формат: **(OR.<booleanArgs>)** или

(XOR.<booleanArgs>) или

(NOT <booleanArg>)

Употреба: задава проста логическа връзка. Конюнкцията (AND) се задава неявно - всяко описание е конюнкция от съответните дескриптори (затова няма нужда от специален дескриптор за конюнкция).

Примери:

```
(OR (a Dog) {(a Cat) (which hasColor
                          Brown)})
(NOT (a Pet with owner =
                       (a Student)))
```

Рестрикция (restriction)

Формат: **(theOne <restrictionDescr>)**

Употреба: маркира съответното описание като достатъчно за означаване (определяне, цитиране) на единствен обект в дадения контекст.

Пример:

```
(theOne {(a Mouse) (which Owns
                        (a Dog))})
```

Селекция (selection)

Формат:

```
(using <selectionDescr>
  selectFrom
    <selectionPattern-1> ~<selection-1>
      ...
    <selectionPattern-n> ~<selection-n>
  otherwise <defaultSelection>)
```

Употреба: декларативна форма, която съответства на операторите от типа на CASE или SELECT в езиците за програмиране.

Описание на множество (set specification)

Формат:

(SetOf <setElementDescr>) или

(In.<setDescr>) или

(Items.<elements>) или

(NotItems.<elements>) или

(AllItems.<elements>) или

(ListOf.<elements>) или

(Sequence.<elements>)

Употреба: тези дескриптори се използват за описание на множества, редици и списъци. Позволяват описанието на даден обект в термините на множеството, към което принадлежи, а също и на дадено множество в термините на обектите, които съдържа то.

Примери:

```
(SetOf {(an Integer) (which hasFactor 2)})
```

(Items 2 4) поне числата 2 и 4 са елементи на множеството

```
(AllItems 2 4 64 {(an Integer)
                  (which hasFactor 3)})
```

(NotItems 5 1) числата 5 и 1 не са елементи на множеството

```
(In {(SetOf (an Integer)) (Items 2 5 8)
     (NotItems 4)})
```

Последният пример представлява описание на обект от множество на цели числа, което включва поне 2, 5 и 8 и не включва 4.

Състояние, случка (contingency)

Формат:

(during <timeSpecification> then
<contingentDescr>)

Употреба: специфицира описание, зависимо от времето (или от някакъв хипотетичен свят).

Пример. Част от описанието на даден конкретен блок от "света на кубовете" може да изглежда по следния начин:

```
(during State24 then (the top from
   (a Stack with height=3)))
```

ПРИМЕР

```
[Person UNIT Basic
   <SELF>
   <firstName (a String)>
   <lastName (a String)>
   <age (an Integer)>
   <homeTown {(a City) PaloAlto ;
                          DEFAULT}>
   <streetAddress (an Address)>]
```

```
[Traveller UNIT Specialization
   <SELF (a Person)>
   <age {(XOR Infant Child Adult)
         (using (the age from Person ThisOne)
            selectFrom
               (which IsLessThan 2) ~Infant
               (which IsGreaterThan 11) ~Adult
            otherwise Child)}>
   <preferredAirport
        {(In (the localAirports from City
              (the homeTown from Person
                 ThisOne)))
         ; DEFAULT
         (an Airport)}>]
```

```
[Airport UNIT Basic
   <SELF>
   <location (a City)>]

[City UNIT Basic
   <SELF>
   <localAirports
      (SetOf (an Airport with
         location = ThisOne)) ;
            DEFAULT>]
```

```
[PaloAlto UNIT Individual
   <SELF (a City with localAirports =
                (Items SJO SFO OAK))>]

[SJO UNIT Individual
     <SELF (an Airport with location =
                SanJose)>]
```

```
[Magnitude UNIT Relation
   <SELF (an ArithmeticRelation)
      (TRIGGERS (ToTest  ... програмен
                  код на LISP ...  ))>
   <greater (a Quantity)>
   <lesser (a Quantity)>]
```

```
[IsGreaterThan PREDICATE Magnitude
    greater lesser]
```

Дефинира предиката IsGreaterThan, като за фокус е избран слотът greater в Magnitude. Аргументът, който следва предиката, се отнася до слота lesser. Този предикат се използва в дескриптори от вида (which IsGreaterThan 2) като еквивалент на (the greater from Magnitude (a Magnitude with lesser=2)).

```
[IsLessThan PREDICATE Magnitude
     lesser greater]
```

Втори предикат, основан на релацията (unit-a) Magnitude, с фокус върху слота lesser.

Пояснения:

- знакът ";" служи за означаване на т. нар. характеристики (features), свързани със съответния слот. По-точно, знакът ";" предшества името на съответната характеристика. Една от тези характеристики, която се използва в нашия пример, е характеристиката DEFAULT. Тя служи за означаване на стойността, която се възприема по премълчаване за съответния слот (ако не е зададена някаква конкретна стойност). В дефиницията на unit-a Person стойността по премълчаване на слота homeTown е PaloAlto (PaloAlto е unit, описан като "a City");

- специалният unit ThisOne се интерпретира като цитиране на обекта, който текущо се описва, когато описанието се използва като прототип;

- дескрипторът (which IsLessThan 2) използва предиката IsLessThan, който сравнява двойки числа и се основава на unit-а Magnitude. Предикатите се дефинират по отношение на (в съответствие с) даден unit, като аргументите на предиката са слотове на unit-а. Два различни предиката са дефинирани в нашия пример в термините на релацията Magnitude. Unit-ът Magnitude съдържа в SELF слота си специална процедура за проверка на верността на отношението (релацията).

Предикатите винаги се използват, като се описва един специфичен аргумент, докато твърденията включват различен брой аргументи (по няколко аргумента), без да се фокусира върху един от тях.

ХАРАКТЕРИСТИКИ (FEATURES) И МЕТАОПИСАНИЯ

Представянето на метазнания в KRL може да става по два начина: чрез използване на т. нар. характеристики (features) и чрез метаописания. Характеристиките са Лисп-ови атоми, а метаописанията са unit-и, които се използват за характеризиране в определен аспект на даден дескриптор (слот).

В примера от предишния параграф показахме как се използва характеристиката DEFAULT (тя служи за означаване на тези дескриптори, които се смятат за валидни при отсъствие на друга (конкретна) информация). В KRL се използват още два типа характеристики, които имат значение за програмите, предназначени за съпоставяне, търсене и добавяне на данни.

Това са характеристиките CRITERIAL (тя служи за означаване на множеството от дескриптори, чието удовлетворяване е критерий за успешно съпоставяне, независимо че освен тези дескриптори в съответното описание може да съществуват и други) и PRIMARY (тя служи за означаване на основната, първичната перспектива за наследяване на свойства при наличие на повече от една йерархия). Възможността за асоцииране на цял unit на мета ниво позволява да се изграждат описания, които включват факти относно други факти (взаимни зависимости между предположения, история на извода или наследяването на даден факт и т.н.).

Тази възможност обаче е все още само хипотетична (теоретична), тъй като в известните реализации на KRL не са включени (и дори не са строго описани в съобщението за езика) съответни стандартни средства.

ТИПОВЕ ПРОЦЕДУРИ, КОИТО СЕ АСОЦИИРАТ КЪМ ОПИСАНИЯТА В KRL

В описанията на различни обекти на KRL могат да се използват разнотипни процедури. Синтактично те винаги са Лисп-ови функции (в оригиналната реализация на KRL: функции на INTERLISP).

Допуска се използване на различни типове процедури. От една страна, процедурите в KRL (както и в стандартните фреймови системи) се делят на демони и методи. От друга страна, говори се още за т. нар. тригери (triggers) и "клопки" (traps). Тригерът е процедура, асоциирана към даден клас (той се прилага към операциите и събитията, които се отнасят до обектите, чието описание включва перспектива, чийто прототип е unit-ът, към който е асоцииран този тригер); trap-ът е процедура, която се отнася до конкретен обект (екземпляр) - той се прилага към обектите и събитията, директно включени в unit-а, към който се отнася този trap.

Към всеки слот от даден unit може да бъде асоцииран списък от съответни тригери и trap-ове. Тригерите и trap-овете могат да бъдат както демони, така и методи.

ОСОБЕНОСТИ НА ПРОЦЕСА НА СЪПОСТАВЯНЕ В KRL

Съпоставяне на множества от описания

Образецът и данните при съпоставянето в KRL могат да съдържат произволен брой дескриптори. За да завърши съпоставянето успешно, е необходимо всички дескриптори в образеца по някакъв начин да се удовлетворяват от данните. За целта matcher-ът на KRL разполага с множество стратегии за сравняване на дескриптори.

Например, ако образецът и данните съдържат дескриптор, който е указател към някакъв индивид, то съответните индивиди се сравняват и съпоставянето успява или пропада в зависимост от това, дали те са идентични. Ако образецът и данните съдържат перспективи с един и същ прототип, то тези перспективи подробно (детайлно) се сравняват.

Алгоритъмът за сравнение взема решения за това, какви подзадачи да бъдат решавани в процеса на съпоставяне и в какъв ред да бъдат решавани тези подзадачи. При това освен вградения в системата (езика) алгоритъм, който се изпълнява по подразбиране, могат да се използват и специални стратегии, дефинирани от потребителя.

Използване на свойства на елементите на данните

Matcher-ът на KRL може да променя нивото, на което съответства "дъното" (простият случай) на съпоставянето. Например, когато сравнява някакъв дескриптор с даден индивид, matcher-ът може да генерира подзадача за съпоставяне (сравняване) на дескриптора от образеца с описанието, което се съдържа в unit-а за този индивид. Това се прави само когато стратегиите за сравняване не могат да открият подходящ дескриптор в данните, но могат да открият указател към индивид.

Използване на вече изведени свойства при съпоставянето

При съпоставянето по принцип се използват свойства, които са зададени в явен вид в базата от знания, но те може да не са локални (да не са зададени локално) за данните, които се съпоставят. В такива случаи често се решават рекурсивни задачи за съпоставяне, при които в крайна сметка се извършва и (логически) извод на свойства.

Допускане и игнориране на частични резултати

Matcher-ът на KRL различава някои случаи на неуспешно (до момента) съпоставяне, като например:

- образецът "демонстративно" (явно, категорично) не е съпоставим със съответните данни;
- съпоставимостта между факта (данните) и образеца не е достатъчно очевидна.

Тези два случая се третират по различен начин (особено тогава, когато съпоставянето се прекрати поради изчерпване на някакъв ресурс или поради изчерпване на известните стратегии за сравнение).

По принцип процесът на съпоставяне в KRL може да бъде прекратен в даден момент и да се върне полученият до този момент частичен резултат. Например, ако след изчерпване на някакъв отпуснат ресурс (дълбочина на изследване и др.) не бъде получен явен положителен или отрицателен резултат, KRL генерира и връща резултат от типа на "don't know yet". Такъв резултат при определени условия може да се тълкува като "yes", но също и като "no" или като причина за продължаване на процеса на съпоставяне, предшествано от отпускане на допълнителни ресурси.

Формиране на частичен (условен) резултат

В случай, когато не е в състояние да даде категоричен резултат, matcher-ът на KRL може да формира частичен (условен) резултат, който да има например следния вид: "Yes, if (a Dog) matches (a Pet)". Този отговор по същество съдържа генерирана от matcher-а подзадача, която по някаква причина не е била решена от него (и от чието решаване зависи крайният резултат).

Използване на редундантна информация

Нека разгледаме следния пример, илюстриращ използването на редундантна информация при изграждането на описания, които ще се използват при съпоставяне (matching). Unit-ът Event234 представя дадено конкретно събитие. Във всички варианти на описание на Event234 получателят в това събитие и обектът на (пре)даване са специфицирани по един и същ начин.

Разглежданите варианти се различават по степента на детайлност, с която е специфицирано другото действащо лице в събитието. При това, вариант 4 се различава съществено от останалите по това, че в него липсва директно цитиране на unit-а, описващ въпросното действащо лице.

Вариант 1

```
[Event234 UNIT Individual
    <SELF (a Give with
              object = (a Pen)
              giver = Person2
              recipient = Person1)>]
```

Вариант 2

```
[Event234 UNIT Individual
   <SELF (a Give with
            object = (a Pen)
            giver = {Person2
                  (which IsHusbandOf
                                 Person3)}
            recipient = Person1)>]
```

Вариант 3

```
[Event234 UNIT Individual
   <SELF (a Give with
            object = (a Pen)
            giver = {Person2
               (which IsHusbandOf
                  {Person3 (a Lawyer)})}
            recipient = Person1)>]
```

Вариант 4

```
[Event234 UNIT Individual
    <SELF (a Give with
              object = (a Pen)
              giver = (which IsHusbandOf
                           (a Lawyer))
              recipient = Person1)>]
```

```
[Give UNIT Specialization
    <SELF (an Event)>
    <object (a Thing)>
    <giver (a Person)>
    <recipient (a Person)>]
```

```
[Lawyer UNIT Specialization
   <SELF (a Person)>]

[Pen UNIT Basic
   <SELF (a PhysicalObject)>]
```

```
[Person1 UNIT Individual
   <SELF (a Person with firstName =
                          "David")>]

[Person2 UNIT Individual
   <SELF {(a Person with firstName =
                          "Jonathan")
          (which IsHusbandOf
                          Person3)}>]

[Person3 UNIT Individual
   <SELF {(a Person with firstName =
                          "Ellen")
          (a Lawyer)}>]
```

Ако опитаме да съпоставим Event234 с образеца

(a Give with giver = (which IsHusbandOf
(a Lawyer))),

усилията за определяне на отговора ще бъдат различни за четирите варианта. В случаите, когато има по-малко редундантна информация, mather-ът трябва да използва и съдържанието на unit-ите Person2 и Person3. Резултатите за различните варианти биха могли и да се различават. *Например*, съпоставяйки (a Give with giver = Person2) и вариант 4, всичко, което може да се каже, е, че те са потенциално сравними, докато за останалите варианти отговорът ще бъде категорично "yes".